Degree in Mathematics

Title: Interval Protection of multi-state tabular data

Author: Anna Via Baraldés

Advisor: Jordi Castro Pérez

Department: Statistics and Operations Research

Academic year:2014-2015

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Abstract

Key words: Linear optimization, Cutting plane methods, Benders decomposition, Large-scale optimization, Statistical disclosure control, Tabular data protection, Interval protection

MSC2000: 90C05, 90C06, 90C25, 90C90

Various methods are used for Statistical Data Protection. The most commonly used method is Cell Supression (CS). This method completely erases the value of the sensitive cells, as well as a set of secondary cells to ensure confidentiality. One of the big disadvantages of this method is the difficulty analyzing the publications that have been protected by CS, since many cells will not have any information at all. This method results in computational problems as well, as many binary variables are used, making the method hard to solve (it has actually been proved to be an NP-hard problem).

An often used alternative to CS is Controlled Tabular Adjustment (CTA). CTA overcomes many of the problems of CS, because the nearest secure values to each cell are published (enabling an easier analysis of the table), and because the number of constraints and variables are significantly less than in CS.

Another method is Interval Protection (IP). By this method, a safe interval is published for each cell, which always covers the true value of the cell. This increments the the usability of the published tabular data protected by Interval Protection.

IP is, in theory, simpler than CTA; as such, IP may be more efficient and provide a better solution than the procedure based on CTA, plus a post-process with one-cell CTA for unprotected cells. In addition, extra constraints can be added to the IP method to force, e.g., that the intervals provided include the adjusted values previously obtained by CTA.

The purpose of this paper is to determine if by applying Benders' Decomposition solving gets more efficient than solving the initial problem as a whole. In order to do this, an AMPL implementation of the Benders' Decomposition applied to the IP problem has been programmed. Several tables have been tested to check the results of this implementation and to see if the final solutions for these tables satisfy the constraints of the problem and therefore are truly safe to publish. Also, the CPU and the number of simplex iterations have been measured in order to compare them with the ones obtained when the problem is solved directly as a whole.

Contents

Chapter 1
Introduction

Governments, agencies and other organizations publish statistics to provide information about a range of topics, for example, economics, health and environment. With these statistics, users are able to know about these topics, and make comparisons between geographical areas, changes over time, and so on. Users can be of any type (governments, research institutions, journalists, firms, etc.) and their goals can be related to business, research, and assessing policies, decisions, or the progress of certain projects.

1. Statistics Dissemination

Almost every country has a National Statistical Agency (NSA) responsible for producing and disseminating the official statistics that citizens need to evaluate operations and policies. Users should be able to form a general view of the information, but should never be able to identify a respondent or a group of respondents, or their responses to the surveys. Due to ethical standards and government legislation, the statistics published have to ensure confidentiality of the information collected. If that were not the case, respondents would not participate in surveys or, if they did, their responses might not be honest.

Examples of current principles and legislation to enforce the confidentiality of published statistics are:

- Principle 6 of the UN Economic Commission report "Fundamental Principles for Official Statistics", April 1992, which states: "Individual data collected by statistical agencies for statistical compilation, whether they refer to natural or legal persons, are to be strictly confidential and used exclusively for statistical purposes."

- Spain's "Ley de la función Estadística Pública" (BOE 11-5-1989, ley 12/1989), articles 12.1 and 12.2, which state: "... serán objeto de protección y quedarán amparados por el secreto estadístico los datos personales que obtengan los servicios estadísticos [...] que, o bien permitan la identificación inmediata de

los interesados, o bien conduzcan por su estructura, contenido o grado de desagregación a la identificación indirecta de los mismos". In Spain, a special agency, the Agencia de Proteccion de datos (AEP), exists. Its purpose is to ensure the fulfillment of the country's data protection regulations.

Disclosure happens when a user can obtain a better estimator about some confidential data because of the published statistic. Following data collection, all forms with full names should be destroyed, but, as we will see, this action alone does not ensure confidentiality. Whenever a statistic is published, some disclosure will happen and confidentiality will be threatened. The only way to avoid full disclosure is not to publish any data, and as such, disclosure can only be limited.

In this sense, we can distinguish between confidential and sensitive data. Confidential data cannot be disseminated, since dissemination will not preserve the respondent's privacy. Sensitive data is not confidential, but it discloses confidential data or makes it possible for users to get a good estimator of that data.

For all these reasons, there are two concerns regarding published statistics:

- confidentiality of the data should be ensured to protect the privacy of the respondents
- data should be beneficial and provide high quality information to users

These two concerns somewhat contradict each other: the safer and better protected some statistic is, the less information we can publish and the greater the information loss.

2. Statistical Data Protection (SDP)

At this stage, Statistical Data Protection (SDP), also known as Statistical Disclosure Control (SDC) or Statistical Disclosure Limitation (SDL) becomes relevant. SDP is a set of tools to ensure that statistical data is disseminated without disclosing the confidential information of the respondents. Through SDP, safely modified data will be published, since the dissemination of the original data would produce disclosure of confidential information, and at the same time it will preserve its utility by producing minimum information loss.

There are basically three types of data:

(1) **Microdata files**: unit record files that contain the responses and attributes of each respondent. Information about each respondent is stored by the statistics agency in these files in order to produce the statistics in a later stage. Formally, a microdata file V of s individuals and t variables is a matrix in $\mathbb{R}^{n\times m}$, in which each row V_i has all the information about respondent i, and each column V_j has the different responses for a particular variable. $V_{i,j}$ is therefore the value of variable j for the individual i.

Formally, a microdata file is a function of this kind:

$$V : I \to D(V_1) \times ... \times D(V_t)$$

The variables can either be numerical or categorical. An example of microdata is the following: out of the inhabitants of Catalonia, store for each one his or her city of birth, his or her profession, and his or her salary. The variables are V_1 city of birth (categorical), V_2 Profession (categorical) and V_3 salary (numerical). For each individual i, $V_{i,1}$ stores its city of birth, $V_{i,2}$ stores its profession, and $V_{i,3}$ its salary.

...	...	...
Barcelona	Doctor	1708.03
Barcelona	Architect	1506.04
Cambrils	Ski Instructor	940.79
...	...	...

Even if the full name of each respondent is erased, there will be risk of disclosure in the case that a respondent has a unique combination of attributes. For example, if there is only a respondent who comes from Cambrils and has the profession of "ski instructor", anybody who knows this person will automatically know his or her salary.

(2) **Tabular data (magnitude or frequency tables)**: Tabular data are created by crossing one or more categorical variables of a microdata file. Tabular data can either be frequency tables (for the crossed variables of each cell, it shows the count of the occurrences of values) or magnitude tables (for the crossed variables of each cell it shows certain information, such us the sum or the mean of a third variable).
Formally, tabular data is a function of this kind:

$$T : D(V_1) \times ... \times D(V_l) \to \mathbb{R}\text{or}\mathbb{N}$$

where $l \leq t$ is the number of crossed categorical variables, and the outcome is $\mathbb{R}$ for magnitude tables, and $\mathbb{N}$ for frequency tables.

An example of frequency table obtained from the previous microdata file would show for each city how many people work in each profession:

	Barcelona	Reus	Cambrils
Doctor	18.000	500	30
Architect	10.000	300	14
Ski Instructor	400	2	1
...	...	...	...

And an example of magnitude table obtained from the previous microdata file, would show for each profession the sum of salaries of the people in each Catalan city:

	Barcelona	Reus	Cambrils
Doctor	36000000	850000	49500
Architect	19000000	360000	20020
Ski Instructor	388000	1600	910
...	...	...	...

Although tabular data contains aggregate data and not individual information, there is still a risk of disclosure of confidential information. We will see an example with the frequency and magnitude tables obtained before. If a cell has a frequency of one (in our case Cambrils and Ski Instructor), anybody can know that the individual has a certain salary (910). This disclosure constitutes an external attack. In the case in which the frequency is two, one individual will be able to know the salary of another individual: if a ski Instructor from Reus has a salary of 700, he or she will automatically know that the other ski Instructor from Reus has a salary of 900. This disclosure constitutes an internal attack. Even in cases in which the frequency is bigger, if one individual's contribution to the cell is relatively much bigger than the others, there is also a risk of disclosure.

Since in tabular data we have a number m of constraints, and for large tabular data this m can be very big, its protection turns out to be either a Linear Optimization or a Mixed Integer Linear Optimization problem, which in many cases is very complex.

(3) **Data obtained from queries to databases**: information retrieved from a database obtained by presenting questions to the database in a predefined format (Structured Query Language SQL is mainly used as the standard query format).

Statistical Data Protection for tabular data has to ensure that all released tabular cells satisfy an appropriate disclosure rule. Cells that fail this rule are sensitive, and protection ranges defined by lower and upper bounds on the true cell value will be assigned to them. No user of the statistics should be able to estimate that the true value of a sensitive cell is within this protection range.

There are some rules that enable us to decide if one cell should be protected (considered sensitive) or can be safely published.

- For frequency tables, we refer to the minimum frequency rule. A cell will be considered sensitive if its frequency is smaller than a certain number t (normally $t = 3$).

- For magnitude tables, the minimum frequency rule is insufficient, since we also have to consider the fact that no individual has a dominant contribution to the cell.
  (1) The rule (n, k) will consider a cell sensitive in the case that n or fewer of the respondents contribute with more than the $k\%$ of the cell's value. Normally it is taken $n = 3$, $k = 70$.
  (2) The rule $p\%$ will consider a cell sensitive in the case that a user can estimate a respondent's value with a precision of $p\%$.

3. Modeling of tabular data

Tabular data can be modelled as:

- Set of n cells a_i, $i \in 1, .., n$ satisfying a set of m linear relationships $Aa = b$, $A \in \mathbb{R}^{nxm}$, $b \in \mathbb{R}^m$.
- If the table is positive, there will also be sign constraints of the type $a_i \geq 0$, $i \in 1, .., .n$.
- Known set of lower and upper bounds l_i, u_i $i \in 1, .., n$ for each cell, i.e. $a_i \in (l_i, u_i)$.
- Weight w_i, related to the cost of modifying or suppressing that cell.
- Parameter in s, u that indicates wether each cell is sensitive.
- Parameters lpl_i, $upl_i \in \mathbb{R}$, which are the lower and upper protection levels for each sensitive cell.

Usually the linear relationships of the matrix only ensure that the last row and column contain the subtotals and totals of each row or column; thus $a_{i,j} \in 1, 0, -1$ (-1 when the cell is a marginal or total cell).

4. Protection Methods

Given the necessity to protect data, there are currently a number of open software options for SDP. Software package Argus has two modules: μ-Argus for microdata files, and τ-Argus for tabular data. There are also special packages for SDP in R. There are two types of SDP methods:

(1) **Perturbative methods**: they publish modified data. Examples of these methods are:
   - **Controlled Rounding Problem (CRP)**: every cell is rounded to the nearest multiple of a number r (usually $r = 5$), with the exception of the totals and subtotals, which have to ensure that the linear relationships of the table are maintained. This algorithm is implemented by τ-argus.
   - **Control Tabular Adjustment (CTA)**: It publishes the closest safe values of the sensitive cells to the original values, and adjusts the rest of the values in order to maintain the linear relationships of the table. This method will publish a new table x such that it minimizes the norm
   $$\|x - a\| = \sum (w_i(x_i - a_i))$$
   ensuring security, since every modified sensitive cell x_i will satisfy $x_i \geq a_i + upl_i$ or $x_i \leq a_i - lpl_i$. This algorithm is also implemented by τ-argus.
   - **Interval Protection (IP)**: A new table is created in which the values a_i of a set of cells H are replaced by an interval $[lbi, ubi]$, such that this interval is as small as possible and $a_i \in [lb_i, ub_i]$. From these intervals, no attacker can determine that $a_i \in (a_i – lpl_i, a_i + upl_i)$ for the sensitive cells.

(2) **Non-perturbative methods**: They do not modify the data but do not fully publish it. Some data is supressed or the table structure is changed. Examples of these methods are:
   - **Recoding**: unifying two categories of the same variable (in our example, we could consider the cities of Cambrils and Reus together) so that the new frequencies are big enough not to be sensitive.
   - **Cell Suppression Problem (CSP)**: Since the suppression of only the sensitive cells is insufficient to ensure confidentiality, as they can be recalculated by the linear relationships of the table, a secondary set of cells to be eliminated has to be determined so that protection will be ensured. Integer Programming will provide a set of secondary cells to be eliminated such that security will be ensured with minimal loss of information. This algorithm is also implemented by τ-argus.

Chapter 2
Interval Protection problem description

We are given a table (i.e., a set of cells $a_i, i \in \mathcal{N} = \{1, \ldots, n\}$), satisfying m linear relations $Aa = b$, $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$). Any set of values x satisfying $Ax = b, l \leq x \leq u$, is a valid table, $l \in \mathbb{R}^n$, $u \in \mathbb{R}^n$ being known a priori lower and upper bounds for cell values. For positive tables we have $l_i = 0,\ a_i = +\infty,\ i = 1, \ldots, n$, but the procedure outlined here is also valid for general tables. We consider that cells provide information about some attribute for several individual states (e.g., member states of the European Union), as well as the highest-level of aggregated information (e.g., at European Union level). The set of multi-state cells, or cells providing this highest-level of aggregated information, will be denoted as $\mathcal{H} \subseteq \mathcal{N}$.

Let $\mathcal{F}, S, M$ be a partition of $\mathcal{N}$, i.e., $\mathcal{N} = \mathcal{F} \cup \mathcal{S} \cup \mathcal{M}$, and $\mathcal{F} \cap \mathcal{S} = \mathcal{F} \cap \mathcal{M} = \mathcal{S} \cap \mathcal{M} = \emptyset$. $\mathcal{S}$ is the set of sensitive cells to be protected, with upper and lower protection levels upl_s and lpl_s for each cell $s \in \mathcal{S}$. $\mathcal{F}$ is the set of (usually state) cells whose values have been previously published by individual states and are thus known. To simplify the formulation of the forthcoming optimization problems, we can assume that for $f \in \mathcal{F}$ we have $l_f = u_f = a_f$, and then cells from $\mathcal{F}$ can be considered elements of $\mathcal{M}$, that is, $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{F}$ and $\mathcal{F} \leftarrow \emptyset$. $\mathcal{M}$ is the set of non-sensitive and non previously published cells. In general, cells in $\mathcal{S}$ provide information at state level, but in some cases multi-state cells may also be sensitive, thus we may have $\mathcal{S} \cap H \neq \emptyset$. Since multi-state cells may not have been previously published, we may also have $\mathcal{M} \cap H \neq \emptyset$. To make the formulation more general our only assumption will be that $\mathcal{H} \subseteq \mathcal{N}$. When $\mathcal{H} = \mathcal{N}$ we have a standard "interval protection" problem (also known as "partial cell suppression", which was its original name).

Our purpose is to publish the set of smallest intervals $[lb_h, ub_h]$—where $l_h \leq lb_h$ and $ub_h \leq u_h$— for each multi-state cell $h \in \mathcal{H}$ instead of the real value $a_h \in [lb_h, ub_h]$, such that, from these intervals, no attacker can determine that $a_s \in (a_s - lpl_s, a_s + upl_s)$ for any sensitive state cells $s \in \mathcal{S}$.

This means that

$$\underline{a_s} \leq a_s - lpl_s \quad \text{and} \quad \overline{a_s} \geq a_s + upl_s, \tag{1}$$

$\underline{a_s}$ and $\overline{a_s}$ being defined as

(2)

$$\underline{a_s} = \begin{array}{lll} \min & x_s & \\ \text{s.to} & Ax = b & \\ & l_i \leq x_i \leq u_i & i \in \mathcal{N} \setminus \mathcal{H} \\ & lb_i \leq x_i \leq ub_i & i \in \mathcal{H} \end{array} \quad \text{and} \quad \overline{a_s} = \begin{array}{lll} \max & x_s & \\ \text{s.to} & Ax = b & \\ & l \leq x \leq u & i \in \mathcal{N} \setminus \mathcal{H} \\ & lb_i \leq x_i \leq ub_i & i \in \mathcal{H} \end{array}$$

Clearly, for cells $i \in \mathcal{H} \cap \mathcal{S}$, (2) implies that $lb_i \leq a_i - lpl_i$ and $ub_i \geq a_i + upl_i$.

The previous problem can be formulated as a large-scale linear optimization problem. For each primary cell $s \in \mathcal{S}$, two auxiliary vectors $x^{l,s} \in \mathbb{R}^n$ and $x^{u,s} \in \mathbb{R}^n$ are introduced to impose, respectively, the lower and upper protection requirement of (1). The problem formulation is as follows:

$$\begin{array}{ll} \min & \displaystyle\sum_{i \in \mathcal{H}} w_i (ub_i - lb_i) \\ \text{s.to} & \end{array}$$

$$\left.\begin{array}{rcll} & Ax^{l,s} & = b & \\ l_i \leq & x_i^{l,s} & \leq u_i & i \in \mathcal{N} \setminus \mathcal{H} \\ lb_i \leq & x_i^{l,s} & \leq ub_i & i \in \mathcal{H} \\ & x_s^{l,s} & \leq a_s - lpl_s & \\ & & & \\ & Ax^{u,s} & = b & \\ l_i \leq & x_i^{u,s} & \leq u_i & i \in \mathcal{N} \setminus \mathcal{H} \\ lb_i \leq & x_i^{u,s} & \leq ub_i & i \in \mathcal{H} \\ & x_s^{u,s} & \geq a_s + upl_s & \end{array}\right\} \quad \forall\, s \in \mathcal{S} \tag{3}$$

$$\begin{array}{l} l_i \leq lb_i \leq a_i \quad i \in \mathcal{H} \\ a_i \leq ub_i \leq u_i \quad i \in \mathcal{H} \end{array}$$

where w_i is a weight for the information loss associated with cell a_i.

Problem (3) is very large (easily in the order of millions of variables and constraints), but it is linear (no binary, no integer variables), and thus theoretically it can be efficiently solved in polynomial time. In practice, the number of constraints can be computationally expensive, but there are ways to deal with them.

Chapter 3
Benders' Decomposition

Benders' Decomposition (named after Jacques F. Benders,[3]) is a method in mathematical programming that allows the solution of large optimization problems where the variables can be separated into two parts. If we consider a problem to have variables (x, w), the interest in considering the problem with the variables separated relies on the fact that one of the parts, w, complicates the problem in the sense that if they are fixed, the solution of the problem is straightforward.

This method is usually used in Mixed Integer Linear Programming (MILP), where the integer variables are the complicating variables. The method solves a series of small problems instead of a single large problem. Given that the computational cost of solving a problem significantly increases with the increase of the number of variables, Benders' decomposition allows us to solve large problems much more efficiently.

Benders' Decomposition method starts with a problem of this kind:

$$\begin{array}{ll} (P)\min & c^T x + d^T w \\ \text{s.to} & A_1 x + A_2 w \geq b \\ & x \geq 0 \\ & w \in W \\ & c, x \in \mathbb{R}^{n_1} d, w \in \mathbb{R}^{n_2}, A_1 \in \mathbb{R}^{mxn_1}, A_2 \in \mathbb{R}^{mxn_2} \end{array}$$

For a fixed $w \in W$, the corresponding problem is:

$$\begin{array}{ll} (Q)\min & c^T x \\ \text{s.to} & A_1 x \geq b - A_2 w \\ & x \geq 0 \\ & c, x \in \mathbb{R}^{n_1}, w \in \mathbb{R}^{n_2}, A_1 \in \mathbb{R}^{mxn_1}, A_2 \in \mathbb{R}^{mxn_2} \end{array}$$

And its dual is the problem:

$$\begin{array}{ll} (Q_D)\max & \lambda^T (b - A_2 w) \\ \text{s.to} & A_1^T \lambda \leq c \\ & \lambda \geq 0 \\ & c, x \in \mathbb{R}^{n_1}, w \in \mathbb{R}^{n_2}, A_1 \in \mathbb{R}^{mxn_1}, A_2 \in \mathbb{R}^{mxn_2} \end{array}$$

Therefore, we can write our initial problem (P) as:

$$\min_w\{d^T w + \min_x\{c^T x : A_1 x \geq (b - A_2 w), x \geq 0\} w \in W\}$$

Dualising the inner minimum we get the equivalent problem:

$$\min_w\{d^T w + \max_\lambda\{\lambda^T (b - A_2 w) : A_1^T \lambda \leq c, \lambda \geq 0\} w \in W\}$$

The key point of this new formulation is that now the feasible region of the inner problem, the dual feasible region $U = \{\lambda s.t. A_1^T \lambda \leq c, \lambda \geq 0\}$ does not depend on the value $w \in W$. If U is empty, then either the primal problem is unbounded or the primal feasible region is also empty, and so the initial problem is infeasible. Assuming U not empty, we can consider $(u_1, ..., u_p), I = 1, ..., p$ the set of extreme points, and $(v_1, ..., v_q), J = 1, ..., q$ the set of extreme rays,

$$U = \{\sum_{i=1}^{p} \alpha_i u^i + \sum_{j=1}^{q} \beta_j v^j s.t. \sum_{i=1}^{q} \alpha_i = 1, \alpha_i \geq 0 \forall i \in I, \beta_j \geq 0 \forall j \in J\}$$

Problem (Q) will be infeasible when problem $(Q_D))$ is unbounded. In this case, the solution λ^* defines an extreme ray. For some $j' \in J = \{1, ..., q\}, v^{j'T}(b - A_2 w) > 0$ and therefore we force $v^{j'T}(b - A_2 w) \leq 0$ in (Q) by adding this new restriction (feasibility cut).

Problem (Q) will be feasible but not optimal when (Q_D) is feasible but not optimal for a given $w \in W$. In this case, the solution $u = \lambda^*$defines an extreme point, and the restriction $z \geq c^T x + u^T(b - Tx)$ is added to the problem (optimality cut).

The original primal problem (P) can be reformulated as follows, which we will call the Master Problem:

$$\begin{array}{ll} (BP)\min & \theta \\ \text{s.to} & \theta \geq d^T w + u^{iT}(b - A_2 w), i = 1, ..., p \\ & v^{jT}(b - A_2 w) \leq 0, j = 1, ..., q \\ & w \in W \end{array}$$

Benders' Decomposition solves a relaxed Master Problem at each iteration, that considers a subset of cuts $I' = \{1, .., k_p\} \subseteq I, J' = \{1, ..k_q\} \subseteq J$. It then solves the dual with the master's fixed solution, and if the optimal is not found, a feasibility or an optimality cut will be added to the Master depending on the solution of the dual. Since I and J are finite, Benders Decomposition method converges after a finite number of steps, and the sequence of solutions of the Master problem converges to the optimal solution of the original problem.

$$\begin{array}{ll} (BPr)\min & \theta \\ \text{s.to} & \theta \geq d^T w + u^{iT}(b - A_2 w), i = 1, ..., k_p \\ & v^{jT}(b - A_2 w) \leq 0, j = 1, ..., k_q \\ & w \in W \end{array}$$

1. Benders' Algorithm

Let (θ_r^*, w_r^*) be the solution of (BPr) and (θ^*, w^*) be the solution of (BP). Initialization: $k_p = 0$ $(I' = \emptyset)$ and $k_q = 0$ $(J' = \emptyset)$. Take $\theta_r^* = -\infty$ and any $w_r^* \in W$.

(1) Solve (Q_D) using fixed $w = w_r^*$.
- If (Q_D) has an optimal solution for this given w, we have an extreme point $u^{i'}$. Because of the relationship between the primal and the dual of a problem,
$$\theta^* \geq d^T w_r^* + u^{i',T}(b - A_2 w_r^*)$$
And since (BPr) is a relaxation of (BP), $\theta_r^* \geq \theta^*$,
  - if $\theta_r^* = d^T w_r^* + u^{i',T}(b - A_2 w_r^*)$, then $\theta_r^* = \theta^*$ and the optimal has been found.BREAK.
  - Otherwise, the constraint $\theta_r^* \geq d^T w_r^* + u^{i',T}(b - A_2 w_r^*)$ is added to (BPr) and $k_p = k_p + 1$.
- If Q_D is unbounded, in the direction $v^{j'}$ from the extreme point $u^{i'}$, a feasibility cut $v^{j',T}(b - A_2 w) \leq 0$ is added to (BPr), and an optimality cut $\theta \geq d^T w + u^{i',T}(b - A_2 w)$ can be added to (BPr) in case the solution does not satisfy this constraint. $k_q = k_q + 1$ and $k_p = k_p + 1$.

(2) If we do not have $w^* = w_r^*$ and $\theta = \theta_r^*$, solve (BPr),get the solution (θ_r^*, w_r^*), and go back to (1).

Chapter 4
Solution of the Interval Protection problem by Benders' Decomposition

Problem (3) has two groups of variables: $x^{l,s} \in \mathbb{R}^n$, $x^{u,s} \in \mathbb{R}^n$; and $lb \in \mathbb{R}^{|\mathcal{H}|}$, $ub \in \mathbb{R}^{|\mathcal{H}|}$, which can be seen as the complicating variables, because if they are fixed the resulting problem in variables $x^{l,s}$ and $x^{u,s}$ is separable, as shown below. Indeed, projecting out the $x^{l,s}$, $x^{u,s}$ variables, (3) can be written as

$$
\begin{array}{ll}
\min & \displaystyle\sum_{i\in\mathcal{H}} w_i(ub_i - lb_i) + Q(ub, lb) \\
\text{s.to} & l_i \le lb_i \le a_i \quad i \in \mathcal{H} \\
 & a_i \le ub_i \le u_i \quad i \in \mathcal{H}
\end{array}
\tag{4}
$$

where

$$
\begin{array}{rll}
Q(ub, lb) = \min & \displaystyle\sum_{s\in\mathcal{S}} ({0_n}^\top x^{l,s} + {0_n}^\top x^{u,s}) = 0 & \\
\text{s.to} & & \\
 & \left.\begin{array}{rcll}
 & Ax^{l,s} & = b & \\
l_i \le & x_i^{l,s} & \le u_i & i \in \mathcal{N}\setminus\mathcal{H} \\
lb_i \le & x_i^{l,s} & \le ub_i & i \in \mathcal{H} \\
 & x_s^{l,s} & \le a_s - lpl_s & \\
 & Ax^{u,s} & = b & \\
l_i \le & x_i^{u,s} & \le u_i & i \in \mathcal{N}\setminus\mathcal{H} \\
lb_i \le & x_i^{u,s} & \le ub_i & i \in \mathcal{H} \\
 & x_s^{u,s} & \ge a_s + upl_s &
\end{array}\right\} & \forall\ s \in \mathcal{S},
\end{array}
\tag{5}
$$

$0_n \in \mathbb{R}^n$ denoting the zero vector. Problem (5) is separable in the $x^{l,s}$, $x^{u,s}$ variables for each $s \in \mathcal{S}$ so it can be replaced by the solution of $2|\mathcal{S}|$ smaller problems of the

form

$$
\begin{array}{rrlll}
Q^{l,s}(ub,lb) = \min & {0_n}^\top x^{l,s} & = 0 & & \\
\text{s.to} & Ax^{l,s} & = b & & \\
l_i \leq & x_i^{l,s} & \leq u_i & & i \in \mathcal{N} \setminus \mathcal{H} \\
lb_i \leq & x_i^{l,s} & \leq ub_i & & i \in \mathcal{H} \\
& x_s^{l,s} & \leq a_s - lpl_s \ , & &
\end{array} \tag{6}
$$

for the lower protection of sensitive cell $s \in \mathcal{S}$, and

$$
\begin{array}{rrlll}
Q^{u,s}(ub,lb) = \min & {0_n}^\top x^{u,s} & = 0 & & \\
\text{s.to} & Ax^{u,s} & = b & & \\
l_i \leq & x_i^{u,s} & \leq u_i & & i \in \mathcal{N} \setminus \mathcal{H} \\
lb_i \leq & x_i^{u,s} & \leq ub_i & & i \in \mathcal{H} \\
& x_s^{u,s} & \geq a_s + upl_s \ . & &
\end{array} \tag{7}
$$

for the upper protection of sensitive cell $s \in \mathcal{S}$.

Denoting the Lagrange multipliers of (6) as $\lambda^{l,s} \in \mathbb{R}^m$ for constraints $Ax^{l,s} = b$; $\mu_u^{l,s} \in \mathbb{R}^n$ for constraints $x_i^{l,s} \leq u_i,\ i \in \mathcal{N} \setminus \mathcal{H}$, and $x_i^{l,s} \leq ub_i,\ i \in \mathcal{H}$ (such that $\mu_{u_i}^{l,s}$ is the multiplier of either type of constraint depending on the particular i); $\mu_l^{l,s} \in \mathbb{R}^n$ for constraints $x_i^{l,s} \geq l_i,\ i \in \mathcal{N} \setminus \mathcal{H}$, and $x_i^{l,s} \geq lb_i,\ i \in \mathcal{H}$; $\nu^{l,s} \in \mathbb{R}$ for the constraint $x_s^{l,s} \leq a_s - lpl_s$; defining

$$
\bar{l}_i = \begin{cases} l_i & \text{if } i \in \setminus \mathcal{H} \\ lb_i & \text{if } i \in \mathcal{N}\mathcal{H} \end{cases} \qquad \bar{u}_i = \begin{cases} u_i & \text{if } i \in \mathcal{N} \setminus \mathcal{H} \\ ub_i & \text{if } i \in \mathcal{H} \end{cases}
$$

and

$$
\tilde{\nu}_i^{l,s} = \begin{cases} \nu^{l,s} & \text{if } i = s \\ 0 & \text{otherwise} \end{cases}
$$

the dual of (6) can be written as

$$
\begin{array}{rrl}
Q_D^{l,s}(ub,lb) = \max & b^\top \lambda^{l,s} + \bar{l}^\top \mu_l^{l,s} - \bar{u}^\top \mu_u^{l,s} - (a_s - lpl_s)\nu^{l,s} \\
\text{s.to} & A^\top \lambda^{l,s} + \mu_l^{l,s} - \mu_u^{l,s} - \tilde{\nu}^{l,s} = 0 \\
& \lambda^{l,s} \text{ free},\ \mu_l^{l,s} \geq 0,\ \mu_u^{l,s} \geq 0, \nu^{l,s} \geq 0,
\end{array} \tag{8}
$$

Similarly, the dual of (7) can be written as

$$
\begin{array}{rrl}
Q_D^{u,s}(ub,lb) = \max & b^\top \lambda^{u,s} + \bar{l}^\top \mu_l^{u,s} - \bar{u}^\top \mu_u^{u,s} + (a_s + upl_s)\nu^{u,s} \\
\text{s.to} & A^\top \lambda^{u,s} + \mu_l^{u,s} - \mu_u^{u,s} + \tilde{\nu}^{u,s} = 0 \\
& \lambda^{u,s} \text{ free},\ \mu_l^{u,s} \geq 0,\ \mu_u^{u,s} \geq 0, \nu^{u,s} \geq 0.
\end{array} \tag{9}
$$

Note that the only difference of formulations (8) and (9) is in the coefficients of $\nu^{l,s}$ and $\nu^{u,s}$, both in the objective function and the constraints, due to the different protection level constraints in (6) and (7).

The feasible region of $Q_D^{l,s}$ is

$$
\mathcal{F}_{l,s} = \left\{ (\lambda^{l,s}, \mu_l^{l,s}, \mu_u^{l,s}, \nu^{l,s}) : \text{satisfy constraints of (8)} \right\} \subseteq \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}.
$$

Then, $\forall \xi_{l,s} \in \mathcal{F}_{l,s}$, $\xi_{l,s} = \sum_{j=1}^{p^{l,s}} \alpha^j z_{l,s}^j + \sum_{j=1}^{q^{l,s}} \beta^j v_{l,s}^j$ with $\alpha^j \geq 0, j = 1, \ldots, p^{l,s}$, $\sum_{j=1}^{p^{l,s}} \alpha^j = 1$, $\beta^j \geq 0, j = 1, \ldots, q^{l,s}$; and $z_{l,s}^1, \ldots, z_{l,s}^{p^{l,s}}$ and $v_{l,s}^1, ..., v_{l,s}^{q^{l,s}}$ being respectively the set of extreme points and extreme rays of $\mathcal{F}_{l,s}$.

Similarly, the feasible region of $Q_D^{u,s}$ is:

$$\mathcal{F}_{u,s} = \{(\lambda^{u,s}, \mu_l^{u,s}, \mu_u^{u,s}, \nu^{u,s}) : \text{satisfy constraints of (9)}\} \subseteq \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R},$$

and then, $\forall \xi_{u,s} \in \mathcal{F}_{u,s}$, $\xi_{u,s} = \sum_{i=1}^{p^{u,s}} \alpha^i z_{u,s}^j + \sum_{j=1}^{q^{u,s}} \beta^j v_{u,s}^j$ with $\alpha^j \geq 0, i = 1, \ldots, p^{u,s}$, $\sum_{i=1}^{p^{u,s}} \alpha^j = 1$, $\beta^j \geq 0, j = 1, \ldots, q^{u,s}$; and $z_{u,s}^1, \ldots, z_{u,s}^{p^{u,s}}$ and $v_{u,s}^1, ..., v_{u,s}^{q^{u,s}}$ being respectively the set of extreme points and extreme rays of $\mathcal{F}_{u,s}$.

Since the objective function of both $Q^{l,s}$ and $Q^{u,s}$ is 0, any feasible solution is optimal. We then have to check whether the current point $(lb, ub) \in \mathbb{R}^{2|H|}$ is optimal for the sub-problems; otherwise feasibility cuts will be added to the master. Therefore, by solving $Q_D^{l,s}$ and $Q_D^{u,s}$ for a fixed $(lb, ub) \in \mathbb{R}^{2|H|}$ we will obtain either a feasible solution or an unbounded solution (that is, $Q^{l,s}$ or $Q^{u,s}$ are infeasible). In the latter case we will have computed an extreme ray $v_{l,s}^i$ or $v_{u,s}^i$ (or both, if both $Q_D^{l,s}$ and $Q_D^{u,s}$ resulted unbounded). Given that we have two sub-problems for each $s \in \mathcal{S}$, for a fixed $(lb, ub) \in \mathbb{R}^{2|H|}$ we can obtain up to $2|S|$ extreme rays v^i that will lead to the addition of up to $2|S|$ feasibility cuts to the master.

Denoting the j-th extreme ray of $Q_D^{l,s}$ as $v_{l,s}^j = (v_j^{\lambda^{l,s}}, v_j^{\mu_l^{l,s}}, v_j^{\mu_u^{l,s}}, v_j^{\nu^{l,s}})$, the feasibility cut to be added to the master problem would be

$$\begin{aligned} 0 \geq \; & b^\top v_j^{\lambda^{l,s}} + \overline{l}^\top v_j^{\mu_l^{l,s}} - \bar{u}^\top v_j^{\mu_u^{l,s}} - (a_s - lpl_s) v_j^{\nu^{l,s}} \\ = \; & \sum_{i=1}^{m} v_{j,i}^{\lambda^{l,s}} b_i + \sum_{i \in \mathcal{N} \setminus \mathcal{H}} (-v_{j,i}^{\mu_u^{l,s}} u_i + v_{j,i}^{\mu_l^{l,s}} l_i) + \sum_{i \in \mathcal{H}} (-v_{j,i}^{\mu_u^{l,s}} ub_i + v_{j,i}^{\mu_l^{l,s}} lb_i) - (a_s - lpl_s) v_j^{\nu^{l,s}} \\ = \; & g_{l,s}^j(ub, lb). \end{aligned}$$

The extreme rays of $Q_D^{u,s}$ have an analogous form $v_{u,s}^j = (v_j^{\lambda^{u,s}}, v_j^{\mu_l^{u,s}}, v_j^{\mu_u^{u,s}}, v_j^{\nu^{u,s}})$ and so does the feasibility cut to be added to the master problem:

$$\begin{aligned} 0 \geq \; & b^\top v_j^{\lambda^{u,s}} + \overline{l}^\top v_j^{\mu_l^{u,s}} - \bar{u}^\top v_j^{\mu_u^{u,s}} + (a_s + upl_s) v_j^{\nu^{u,s}} \\ = \; & \sum_{i=1}^{m} v_{j,i}^{\lambda^{u,s}} b_i + \sum_{i \in \mathcal{N} \setminus \mathcal{H}} (-v_{j,i}^{\mu_u^{u,s}} u_i + v_{j,i}^{\mu_l^{u,s}} l_i) + \sum_{i \in \mathcal{H}} (-v_{j,i}^{\mu_u^{u,s}} ub_i + v_{j,i}^{\mu_l^{u,s}} lb_i) + (a_s + upl_s) v_j^{\nu^{u,s}} \\ = \; & g_{u,s}^j(ub, lb). \end{aligned}$$

The Benders' Master problem is thus

$$\begin{array}{ll} \min & \sum_{i \in \mathcal{H}} w_i (ub_i - lb_i) \\ \text{s.to} & g_j^{l,s}(ub, lb) \leq 0, j \in \{1, \ldots, q^{l,s}\}, \; s \in \mathcal{S} \\ & g_j^{u,s}(ub, lb) \leq 0, j \in \{1, \ldots, q^{u,s}\}, \; s \in \mathcal{S} \\ & l_i \leq lb_i \leq a_i, i \in \mathcal{H} \\ & a_i \leq ub_i \leq u_i, i \in \mathcal{H}. \end{array} \tag{10}$$

Denoting as $\mathcal{I}_{l,s}$ and $\mathcal{I}_{u,s}$ the set of indexes of feasibility cuts obtained from $Q_D^{l,s}$ and $Q_D^{u,s}$, the restricted master problem is:

$$
\begin{array}{ll}
\min & \sum_{i\in\mathcal{H}} w_i(ub_i - lb_i) \\
\text{s.to} & g_j^{l,s}(ub, lb) \leq 0, j \in \mathcal{I}_{l,s} \subseteq \{1, \ldots, q^{l,s}\} \\
 & g_j^{u,s}(ub, lb) \leq 0, j \in \mathcal{I}_{u,s} \subseteq \{1, \ldots, q^{u,s}\} \\
 & l_i \leq lb_i \leq a_i, i \in \mathcal{H} \\
 & a_i \leq ub_i \leq u_i, i \in \mathcal{H}.
\end{array} \tag{11}
$$

The Benders' Decomposition algorithm will then solve (2) for the restricted master problem and (8) and (9) for the sub-problems.

Chapter 5
Implementation

In this chapter we are going to explain how the implementation of the algorithm works, using a simple example.

We consider the following simple table:

10	15	25
20	17	37

This table has $n = 6$ cells, satisfying $m = 2$ linear constraints,

$$a_1 + a_2 - a_3 = 0$$

$$a_4 + a_5 - a_6 = 0$$

We are considering the set of multi-state cells $\mathcal{H} = \mathcal{N} = \{1, 2, 3, 4, 5, 6\}$. There are two sensitive cells a_1 and a_5, which satisfy:

s	p_s	$a_{p[s]}$	lpl_s	upl_s
1	1	10	5	5
2	5	17	7	4

(1) **Initialization**
The number of cuts for the lb variables and the ub variables is set to 0, this means $nCUTls = nCUTus = 0$. The Master Problem is solved, with no feasiblity cuts. Therefore, the problem that is being solved is:

$$\begin{array}{ll} \min & \sum_{i=1}^{6}(ub_i - lb_i) \\ \text{s.to} & l_i \leq lb_i \leq a_i, i \in \{1, 2, 3, 4, 5, 6\} \\ & a_i \leq ub_i \leq u_i, i \in \{1, 2, 3, 4, 5, 6\} \end{array}$$

The values of this Master Problem's solution, lb and ub, will be the fixed values of lb and ub used in the subproblems $QD_{ls}(lb, ub)$ and $QD_{us}(lb, ub)$ in the first iteration.

(2) **Iterating through Benders' algorithm**
As long as an optimal feasible solution for the original problem is not found, the dual subproblems $QD_{ls}(lb, ub)$ and $QD_{us}(lb, ub)$ are solved $\forall s \in \mathcal{S}$. If one of them is unbounded, its corresponding feasibility cut is added to the Master Problem. The algorithm will stop when for some fixed lb and ub all

dual subproblems are not unbounded, and this will be the final solution of the Master Problem.

- **Iteration 1** $\forall s \in S$ the problems $QD_{l,s}$ and $QD_{u,s}$ are solved.

  - **s = 1**

    $QD_{l,1}$ is unbounded. The extreme ray $v^1_{l,s} = (v_1^{\lambda^{l,s}}, v_1^{\mu_l^{l,s}}, v_1^{\mu_u^{l,s}}, v_1^{\nu^{l,s}})$ is obtained: $v_1^{\lambda^{l,s}} = 0$,

    $$v_1^{\mu_l^{l,s}} = \begin{cases} 1, i = 1 \\ 0, \text{otherwise} \end{cases}$$

    $v_1^{\mu_u^{l,s}} = 0$ and $v_1^{\nu^{l,s}} = 1$. The feasibility cut is calculated:

$$0 \geq b^\top v_j^{\lambda^{l,s}} + \bar{l}^\top v_j^{\mu_l^{l,s}} - \bar{u}^\top v_j^{\mu_u^{l,s}} - (a_{p_1} - lpl_s) v_j^{\nu^{l,s}}$$
$$= 0 + \bar{l}_1 v_{1,1}^{\mu_l^{l,s}} + 0 - (a_{p_1} - lpl_1) v_1^{\nu^{l,s}} = 1lb_1 - (10-5)1 = lb_1 - 5$$

    And so the constraint $lb_1 \leq 5$ is added to the Master.

    $QD_{u,1}$ is also unbounded. The extreme ray $v^1_{u,s} = (v_1^{\lambda^{u,s}}, v_1^{\mu_l^{u,s}}, v_1^{\mu_u^{u,s}}, v_1^{\nu^{u,s}})$ is obained: $v_1^{\lambda^{u,s}} = 0$, $v_1^{\mu_l^{u,s}} = 0$,

    $$v_1^{\mu_u^{u,s}} = \begin{cases} 1, i = p_1 = 1 \\ 0, \text{ otherwise} \end{cases}$$

    and $v_1^{\nu^{l,s}} = 1$. The feasibilty cut is calculated:

$$0 \geq b^\top v_1^{\lambda^{u,s}} + \bar{l}^\top v_1^{\mu_l^{u,s}} - \bar{u}^\top v_1^{\mu_u^{u,s}} + (a_{p_1} + upl_1) v_1^{\nu^{u,s}}$$
$$= 0 + 0 + \bar{u}_1 v_{1,1}^{\mu_u^{u,s}} + (a_{p_1} + upl_1) v_1^{\nu^{u,s}} = 1ub_1 + (10+5)1 = ub_1 + 15$$

    The cut that is added to the Master is $ub_1 \geq 15$.

  - **s=2**

    $QD_{l,2}$ is unbounded. The extreme ray obtained is $v^2_{l,s} = (v_2^{\lambda^{l,s}}, v_2^{\mu_l^{l,s}}, v_2^{\mu_u^{l,s}}, v_2^{\nu^{l,s}})$, with $v_2^{\lambda^{l,s}} = 0$,

    $$v_2^{\mu_l^{l,s}} = \begin{cases} 1, i = 5 (= p_2) \\ 0, \text{ otherwise} \end{cases}$$

    $v_2^{\mu_u^{l,s}} = 0$ and $v_2^{\nu^{l,s}} = 1$. The feasibility cut is calculated in an analogous way as in the subproblem $QD_{l,1}$ obtaining $0 \geq -10 + 1lb_5$. The constraint $lb_5 \leq 10$ is added to the Master.

    $QD_{u,2}$ is unbounded. The extreme ray obtained is $v^1_{u,s} = (v_2^{\lambda^{u,s}}, v_2^{\mu_l^{u,s}}, v_2^{\mu_u^{u,s}}, v_2^{\nu^{u,s}})$, with $v_2^{\lambda^{u,s}} = 0$, $v_2^{\mu_l^{u,s}} = 0$,

    $$v_2^{\mu_u^{u,s}} = \begin{cases} 1, i = 5 (= p_2) \\ 0, \text{ otherwise} \end{cases}$$

    and $v_2^{\nu^{l,s}} = 1$.
    Analogous to the calculation of the feasibility cut in $QD_{u,1}$, the cut $ub_2 \geq 15$ is obtained and added to the Master.

- **Iteration 2**
  Four feasibility cuts have been added to the Master ($nCUTls = nCUTus = 2$), and so the resulting problem is:

$$\begin{array}{ll} \min & \sum_{i=1}^{6}(ub_i - lb_i) \\ \text{s.to} & l_i \leq lb_i \leq a_i, i \in \{1,2,3,4,5,6\} \\ & a_i \leq ub_i \leq u_i, i \in \{1,2,3,4,5,6\} \\ & lb_1 \leq 5 \\ & ub_1 \geq 15 \\ & lb_5 \leq 10 \\ & ub_5 \geq 21 \end{array}$$

  The cuts that have been added are the obvious cuts that derive from (2). The Master Problem is solved and its' solution is: $lb = [5, 15, 25, 20, 10, 37]$ and $ub = [15, 15, 25, 20, 21, 37]$. Now the dual subproblems have to be solved again with these new fixed ub and lb.

  - **s=1**
    $QD_{l,1}$ is unbounded. The extreme ray obtained is:

$$v_3^{\lambda_l^{l,s}} = \begin{cases} 1, i = 1 \\ 0, \text{ otherwise} \end{cases}$$

$$v_3^{\mu_l^{l,s}} = \begin{cases} 1, i = 3 \\ 0, \text{ otherwise} \end{cases}$$

$$v_3^{\mu_u^{l,s}} = \begin{cases} 1, i = 2 \\ 0, \text{ otherwise} \end{cases}$$

    and $v_3^{\nu^{l,s}} = 1$. The feasiblity cut is calculated by $0 \geq -5 + lb_3 - ub_2$ and so the cut $lb_3 - ub_2 \leq 58$ is added to the Master.

    $QD_{u,1}$ is also unbounded. For this subproblem the extreme ray obtained is

$$v_3^{\lambda_u^{u,s}} = \begin{cases} 1, i = 1 \\ 0, \text{ otherwise} \end{cases}$$

$$v_3^{\mu_l^{u,s}} = \begin{cases} 1, i = 2 \\ 0, \text{ otherwise} \end{cases}$$

$$v_3^{\mu_u^{u,s}} = \begin{cases} 1, i = 3 \\ 0, \text{ otherwise} \end{cases}$$

    and $v_3^{\nu^{l,s}} = 1$. And so the cut $lb_2 - ub_2 \geq 15$ is added to the Master.

  - **s = 2**
    $QD_{l,4}$ is ubounded, the extreme ray obtained is,

$$v_4^{\lambda_l^{l,s}} = \begin{cases} 1, i = 2 \\ 0, \text{ otherwise} \end{cases}$$

$$v_4^{\mu_l^{l,s}} = \begin{cases} 1, i = 6 \\ 0, \text{ otherwise} \end{cases}$$

$$v_4^{\mu_u^{l,s}} = \begin{cases} 1, i = 4 \\ 0, \text{ otherwise} \end{cases}$$

and $v_4^{\nu^{l,s}} = 1$. The cut that is added to the master is $lb_6 - ub_4 \leq 21$.

$QD_{u,2}$ is unbounded. The extreme ray obtained is $v_4^{\lambda^{u,s}} = 0$, $v_4^{\mu_l^{u,s}} = 0$,

$$v_4^{\lambda_u^{u,s}} = \begin{cases} -1, i = 2 (= p_4) \\ 0, \text{ otherwise} \end{cases}$$

$$v_4^{\mu_l^{u,s}} = \begin{cases} 1, i = 4 \\ 0, \text{ otherwise} \end{cases}$$

$$v_4^{\mu_u^{u,s}} = \begin{cases} 1, i = 6 \\ 0, \text{ otherwise} \end{cases}$$

and $v_4^{\nu^{l,s}} = 1$. And so the last feasibility cut $ub_6 - lb_4 \geq 39$ is added to the master.

- **Iteration 3**
  At the end of iterating through $\mathcal{S}$, $nCUTls = 4$ and $nCUTus = 4$, which means that 8 feasibility cuts have been added to the Master Problem (4 in the first Benders' iteration and 4 more in the next one), which now consists of:

$$\begin{array}{ll} \min & \sum_{i=1}^{6}(ub_i - lb_i) \\ \text{s.to} & l_i \leq lb_i \leq a_i, i \in \{1,2,3,4,5,6\} \\ & a_i \leq ub_i \leq u_i, i \in \{1,2,3,4,5,6\} \\ & lb_1 \leq 5 \\ & ub_1 \geq 15 \\ & lb_5 \leq 10 \\ & ub_5 \geq 21 \\ & lb_3 - ub_2 \leq 58 \\ & lb_2 - ub_2 \geq 15 \\ & lb_6 - ub_4 \leq 21 \\ & ub_6 - lb_4 \geq 39 \end{array}$$

  Its solution is: $lb = [5, 15, 20, 16, 10, 30]$ and $ub = [15, 15, 30, 20, 21, 37]$. Again, with these fixed lb and ub the dual subproblems are solved. In this case, all four dual subproblems are bounded, and so, the previous solution lb ub of the Master Problem was already feasible for the original problem. Therefore the optimal solution for the Interval Protection Problem has been found, with $\sum_{i=1}^{6}(ub_i - lb_i) = 42$.

(3) **Auditing** To ensure that this solution satifies that no attacker can determine that $a_{p(s)} \in (a_{p(s)} - lpl_s, a_{p(s)} + upl_s)$ for $s \in \{1, 2\}$, the problems (2) are solved. In this case, $\underline{a_1} = 5$, $\overline{a_1} = 15$, $\underline{a_5} = 10$ and $\overline{a_5} = 21$. Therefore, it can be asserted that it is safe to publish this solution.

(4) **Solving the initial problem** In this example, since the number of variables is small, the initial problem can be solved directly, without use of Benders' Decomposition. By solving the original problem (3), we obtain a solution with $\sum_{i=1}^{6}(ub_i - lb_i) = 42$. Therefore, it can be asserted that the solution found by Benders' Decomposition is indeed optimal.

(5) **Publication of the table**
It has been proved that the smallest intervals that can be safely published, instead of the values a_i are:

[5,15]	[15,15]	[20,30]
[16,20]	[10,21]	[30,37]

Chapter 6
Computational results

We have applied the AMPL implementation of the Benders' Decomposition for the IP problem (Apendix A: BendersIP.mod and BendersIP.run) to different tables. To compare the efficiency of this implementation, we have also solved the same tables with the direct solving of problem (3) (Apendix B: directe.mod and directe.run).

Denoting n as the number of cells, p as the number of sensible cells and m as the number of constraints, the tables that have been considered satisfy the following:

table	n	p	m
targus	162	13	63
table1	121	10	55
table2	1680	158	299
table3	600	53	170
table4	756	68	243
table5	168	14	62
table6	1584	143	485

The results obtained after applying the Benders' Decomposition implementation for the IP problem to these tables are the following, where CPU is the CPU seconds used by all solve commands, it_B is the number of Benders' iterations, it_s is the total number of simplex iterations (considering the solving of all the dual subproblems and the master problems) and obj is the value of the objective function in the optimal solution:

table	CPU	it_B	it_S	obj
targus	5.16821	31	8872	2142265.7
table1	3.40848	26	7167	136924
table2	410.531	43	1104884	43715149
table3	26.397	43	131834	3624906
table4	50.9263	33	144963	9134139
table5	3.9504	19	5959	303844
table6	966.281	70	1729767	21302104

Here follows the results obtained after applying the direct implementation for the IP problem to these tables. CPU denotes the CPU seconds used by the solve

command, it_S the number of simplex iterations, n_v the number of variables involved ($2ns$) and obj the value of the objective function. There are some problems that we have aborted after a long time of computation without having arrived to an optimal solution; these are denoted as "ab", and their corresponding obj is the value the objective function had in the moment of the program was aborted.

table	*CPU*	it_S	n_v	*obj*
targus	36.0515	16532	4212	2142265.7
table1	3.43548	7452	2420	136924
table2	ab:2944.87	-	530880	16056608400
table3	ab:522.875	-	63600	260592812
table4	11085.6	436895	102816	9134139
table5	10.6764	17325	4704	303844
table6	ab:7816.61	-	453024	4404161015

It can be seen how for all these tables both the CPU and the total number of Simplex iterations is smaller when the problem is solved with Benders' Decomposition than when solved directly. Therefore, Benders' Decomposition makes the solving of the IP problem much more efficient for large tables.

Chapter 7
Conclusion

By Benders' Decomposition method, smaller subproblems of the initial IP problem have been considered and solved. As such, the computational costs should have been significantly less expensive than solving the large problem as a whole.

Solving the IP problem directly would imply to solve a problem of $2sn$ variables (where n is the number of cells and s is the number of sensitive cells). Therefore for problems with $n = 1e5$ and $s = 1e3$, the number of variables involved would be $1e8$. It is for these large tables that Bender's Decomposition implementation for the IP problem would make the solving much more efficient.

Looking at the computational results, it has been proved that indeed for medium and large tables, the Benders' Decomposition method provides a solution to the problem with much less CPU and much less number of simplex iterations. It can be therefore asserted that Benders' Decomposition is an efficient method to solve the IP problem.

Appendix

Here follows the AMPL implementation of the Benders' Decomposition method for the Interval Protection problem. We are considering $H = N$, so that the final result is a set of intervals consisting on an interval for each cell of the table. In the case a group of cells was wanted to be publish without modification, it would be enough to set $u[j] = l[j] = a[j]$, for all cells in this group.

- BendersIP.mod

```
################################################
# PARAMETERS
################################################

param ncells integer >0; # number of cells

param npcells integer>0; # number of sensible cells

param nnz integer >0;

param nconstraints integer >0; # number of restrictions (m)

param c {1..ncells} ; # weights of the cells (w_i)

param is_p {1..ncells}; # 1 if the cell is sensible, 0 otherwise
param is_h {1..ncells}; # 1 if the cell is multi-state, 0 otherwise

param a {1..ncells}; # original values of the cells
param l {1..ncells}; # lower bounds for increments/decrements
param u {1..ncells}; # upper bounds for increments/decrements
param b {1..nconstraints}; # right hand side for constraints

param begconst{1..nconstraints+1}; #pointer to begin info. in coef and xcoef
param coef{1..nnz}; #constraints coefficients
param xcoef{1..nnz}; #index of variable affected for each coefficient

param plpl {1..npcells}; # lower protection limit of primary/sensitive cells (lpl_i)
param pupl {1..npcells}; # upper protection limit of primary/sensitive cells (upl_i)

param p {1..npcells}; # position of primary/sensitive cells

param Trasbegconst{1..ncells+1}; #pointer to begin info. in Trascoef and Trasxcoef
param Trascoef{1..nnz}; #constraints coefficients
param Trasxcoef{1..nnz}; #index of variable affected fore each coefficient

param Ep default .000001; #maximum relative deviation allowed

param stop;
param it;
```

```
################################################
# Definition of the lower subproblem QD_ls #
################################################

var la_ls {1..nconstraints};
var mu_l_ls {1..ncells} >= 0;
var mu_u_ls {1..ncells} >= 0;
var v_ls >= 0;

param lbsub{1..ncells};
param ubsub{1..ncells};

param s integer >0;

param rls;

maximize QD_ls:
	-rls*v_ls + sum {i in 1..nconstraints} (b[i]*la_ls[i]) + sum {i in 1..ncells}
(lbsub[i]*mu_l_ls[i]-ubsub[i]*mu_u_ls[i]) ;
subj to R_QD_ls1 {i in 1..ncells : i != p[s]}:
	sum{t in Trasbegconst[i]..Trasbegconst[i+1]-1} Trascoef[t]*la_ls[Trasxcoef[t]] + mu_l_ls[i]
- mu_u_ls[i] = 0 ;
subj to R_QD_ls2:
	-v_ls + sum{t in Trasbegconst[p[s]]..Trasbegconst[p[s]+1]-1}
Trascoef[t]*la_ls[Trasxcoef[t]] + mu_l_ls[p[s]] - mu_u_ls[p[s]] = 0 ;

#####################################################
# Definition of the upper subproblem QD_us #
#####################################################

var la_us {1..nconstraints};
var mu_l_us {1..ncells} >= 0;
var mu_u_us {1..ncells} >= 0;
var v_us >= 0;

param rus;

maximize QD_us:
	rus*v_us + sum {i in 1..nconstraints} (b[i]*la_us[i]) + sum {i in 1..ncells}
(lbsub[i]*mu_l_us[i]-ubsub[i]*mu_u_us[i]) ;
subj to R_QD_us1 {i in 1..ncells : i != p[s]}:
	sum{t in Trasbegconst[i]..Trasbegconst[i+1]-1} Trascoef[t]*la_us[Trasxcoef[t]] +
mu_l_us[i] - mu_u_us[i] = 0;
subj to R_QD_us2:
	v_us + sum{t in Trasbegconst[p[s]]..Trasbegconst[p[s]+1]-1}
Trascoef[t]*la_us[Trasxcoef[t]] + mu_l_us[p[s]] - mu_u_us[p[s]] = 0;

################################################### #
# Definition of the master problem #
###################################################

param nCUTls >= 0 integer;
```

```
param nCUTus >= 0 integer;

param iter >= 0 integer;
param mipgap;

param constls {1..nCUTls} default 0;
param constlsu {1..ncells,1..nCUTls} default 0;
param constlsl {1..ncells,1..nCUTls} default 0;

param constus {1..nCUTus} default 0;
param constusu {1..ncells,1..nCUTus} default 0;
param constusl {1..ncells,1..nCUTus} default 0;

var lb {1..ncells};
var ub {1..ncells};


minimize BPr: sum{i in 1..ncells}(c[i]*(ub[i]-lb[i]));
#Feasible cuts
subj to Cut_Pointls {j in 1..nCUTls}:
        0 >= constls[j] - sum{i in 1..ncells}(constlsu[i,j]*ub[i]) + sum{i in 1..ncells}
(constlsl[i,j]*lb[i]);
subj to Cut_Pointus {j in 1..nCUTus}:
        0 >= constus[j] - sum{i in 1..ncells}(constusu[i,j]*ub[i]) + sum{i in 1..ncells}
(constusl[i,j]*lb[i]);
#Master restrictions
subj to R1 {i in 1..ncells}:
        lb[i] >= l[i];
subj to R2 {i in 1..ncells}:
        lb[i] <= a[i];
subj to R3 {i in 1..ncells}:
        ub[i] <= u[i];
subj to R4 {i in 1..ncells}:
        ub[i] >= a[i];

#########################################################
# Definition of Q_l,s #
#########################################################

var x_ls{1..ncells};

minimize BQ_ls:
        sum{i in 1..ncells}x_ls[i]*0;
subj to QR1{i in 1..ncells}:
        x_ls[i] >= lbsub[i];
subj to QR2{i in 1..ncells}:
        x_ls[i] <= ubsub[i];
subj to QR3{i in 1..ncells}:
        l[i] <= x_ls[i];
subj to QR4{i in 1..ncells}:
        x_ls[i] <= u[i];
subj to QR5:
```

```
        x_ls[p[s]] <= a[p[s]] - plpl[s];
subj to QR6 {i in 1..nconstraints}:
        sum {t in begconst[i]..begconst[i+1]-1} coef[t]*x_ls[xcoef[t]] = b[i];


#########################################################
# Definition of Q_u,s #
#########################################################

var x_us{1..ncells};

minimize BQ_us:
        sum{i in 1..ncells}x_us[i]*0;
subj to QRu1{i in 1..ncells}:
        x_us[i] >= lbsub[i];
subj to QRu2{i in 1..ncells}:
        x_us[i] <= ubsub[i];
subj to QRu3{i in 1..ncells}:
        l[i] <= x_us[i];
subj to QRu4{i in 1..ncells}:
        x_us[i] <= u[i];
subj to QRu5:
        x_us[p[s]] >= a[p[s]] + pupl[s];
subj to QRu6 {i in 1..nconstraints}:
        sum {t in begconst[i]..begconst[i+1]-1} coef[t]*x_us[xcoef[t]] = b[i];
```

- BendersIP.run

```
reset;

model BendersIP.mod;
data small.txt;

option solver cplexamp;
option omit_zero_rows 1;
option display_eps .000001;
option cplex_options $cplex_options 'presolve= 0';

problem Master: lb,ub,Cut_Pointls,Cut_Pointus,R1,R2,R3,R4,BPr;
problem Q_ls: x_ls, QR1, QR2, QR3, QR4,QR5,QR6,BQ_ls;
problem Q_us: x_us, QRu1, QRu2, QRu3, QRu4, QRu5, QRu6, BQ_us;
problem SubQD_ls: la_ls,mu_l_ls,mu_u_ls,v_ls,R_QD_ls1,R_QD_ls2,QD_ls;
problem SubQD_us: la_us,mu_l_us,mu_u_us,v_us,R_QD_us1,R_QD_us2,QD_us;

suffix unbdd OUT;

############################################################## # # #
#Reading of data and initializations # #
##############################################################

printf"\n Size of the problem \n\n" >output.txt;
printf"\t Number of cells = %d\n",ncells >output.txt;
printf"\t Number of sensible cells = %d\n",npcells >output.txt;
printf"\t Number of constraints = %d\n",nconstraints >output.txt;
printf"\t Numero of non-zeros in the constraints = %d\n",nnz >output.txt;
############################################################## # #
#Benders #
##############################################################
let nCUTls := 0;
let nCUTus := 0;
let it := 1;
let mipgap := 0.01;
let stop := 0;
#let Theta := MinTheta;
#let QDBest:=Infinity;

repeat
{
printf "\nMASTER ITERATION %d\n\n", it >output.txt;
printf "\nSOLVING MASTER PROBLEM\n\n" >output.txt;
problem Master;
display nCUTus >output.txt;
display nCUTls >output.txt;
option cplex_options 'mipdisplay=1 timing=1 feasibility=1.0e-9 integrality=0 primalopt';
solve Master;

if (stop == 1) then printf "\n FINAL SOLUTION\n\n" >output.txt;
for{i in 1..ncells}
```

```
{
let lbsub[i] := lb[i];
}
for{i in 1..ncells}
{
let ubsub[i] := ub[i];
}
display lbsub >output.txt;
display ubsub >output.txt;
if  (stop == 1) then break;
let stop := 1;
let it := it + 1;

for {s2 in 1..npcells}
{
	printf "\n SUBPROBLEM s = %d\n\n", s2 >output.txt;
	#Recalculate parameters
	let s := s2;
	let rls := a[p[s2]]-plpl[s2];
	let rus := a[p[s2]]+pupl[s2];
	problem SubQD_ls;
	option cplex_options 'presolve=0 mipdisplay=1 timing=1 feasibility=1.0e-9 integrality=0
primalopt';
	#solve Q_ls;
	if Q_ls.result = "infeasible" then
	{
		print "Problem Q_ls INFEASIBLE\n";
	}
	else if Q_ls.result = "unbounded" then
	{
		print "Problem Q_ls UNBOUNDED\n";
	}
	solve SubQD_ls; # Solving  (Qd_ls) for fixed lb ub
	if SubQD_ls.result = "unbounded" then
	{
		printf "UNBOUNDED QD_ls\n" >output.txt;
		let nCUTls := nCUTls + 1;

		let constls[nCUTls] := sum{i in 1..nconstraints}(b[i]*la_ls[i].unbdd) -
rls*v_ls.unbdd;
		display constls[nCUTls] >output.txt;

		for {i in 1..ncells} #calculate constlsu[i,j]
		{
			let constlsu[i,nCUTls] := mu_u_ls[i].unbdd;
		}

		for {i in 1..ncells} #calculate constlsl[i,j]
		{
			let constlsl[i,nCUTls] := mu_l_ls[i].unbdd;
		}
		display la_ls.unbdd >output.txt;
```

```
		display mu_l_ls.unbdd >output.txt;
		display mu_u_ls.unbdd >output.txt;
		display v_ls.unbdd >output.txt;
		let stop := 0;
	}
	else printf "OPTIMAL QD_ls\n" >output.txt;
	#solve Q_us;
	if Q_us.result = "infeasible" then {
		printf "Problem  Q_us INFEASIBLE \n";
	}
	if Q_us.result = "unbounded" then {
		printf "Problem Q_ls UNBOUNDED\n";
	}
	solve SubQD_us; #Solving (QD_us) for fixed ub lb
	if SubQD_us.result = "unbounded" then
	{
		printf "UNBOUNDED QD_us \n" >output.txt;
		let nCUTus := nCUTus + 1;

		let constus[nCUTus] := sum{i in 1..nconstraints}(b[i]*la_us[i].unbdd) +
rus*v_us.unbdd;
		display constus[nCUTus] >output.txt;
		for {i in 1..ncells} #calculate constusu[i,j]
		{
			let constusu[i,nCUTus] := mu_u_us[i].unbdd;
		}
		for {i in 1..ncells} #calculate constusl[i,j]
		{
			let constusl[i,nCUTus] := mu_l_us[i].unbdd;
		}

	display la_us.unbdd >output.txt;
	display mu_l_us.unbdd >output.txt;
	display mu_u_us.unbdd >output.txt;
	display v_us.unbdd >output.txt;
	let stop := 0;
	}
	else printf "OPTIMAL QD_us\n" >output.txt;



}
};
```

Here follows the implementation of the direct solving (without Benders' Decomposition) for the Interval Protection problem.

- IP.mod

```
################################################
# PARAMETERS
################################################

param ncells integer >0; # number of cells

param npcells integer>0; # number of sensible cells

param nnz integer >0;

param nconstraints integer >0; # number of restrictions (m)

param c {1..ncells} ; # weights of the cells (w_i)

param is_p {1..ncells}; # 1 if the cell is sensible, 0 otherwise

param a {1..ncells}; # original values of the cells
param l {1..ncells}; # lower bounds for increments/decrements
param u {1..ncells}; # upper bounds for increments/decrements
param b {1..nconstraints}; # right hand side for constraints

param begconst{1..nconstraints+1}; #pointer to begin info. in coef and xcoef
param coef{1..nnz}; #constraints coefficients
param xcoef{1..nnz}; #index of variable affected for each coefficient

param plpl {1..npcells}; # lower protection limit of primary/sensitive cells (lpl_i)
param pupl {1..npcells}; # upper protection limit of primary/sensitive cells (upl_i)

param p {1..npcells}; # position of primary/sensitive cells

param Trasbegconst{1..ncells+1}; #pointer to begin info. in Trascoef and Trasxcoef (Matriu
Trasposada)
param Trascoef{1..nnz}; #constraints coefficients (Matriu Trasposada)
param Trasxcoef{1..nnz}; #index of variable affected fore each coefficient (Matriu Trasposada)

param Ep default .000001; #maximum relative deviation allowed
```

```
#####################################################
# Definition of the problem #
#####################################################

param mipgap;


var lb {1..ncells};
var ub {1..ncells};
var x_ls {1..ncells,1..npcells};
var x_us {1..ncells,1..npcells};

minimize BPr: sum{i in 1..ncells}(c[i]*(ub[i]-lb[i]));

subj to R1 {i in 1..ncells}:
        lb[i] >= l[i];
subj to R2 {i in 1..ncells}:
        lb[i] <= a[i];
subj to R3 {i in 1..ncells}:
        ub[i] <= u[i];
subj to R4 {i in 1..ncells}:
        ub[i] >= a[i];

subj to QR1{i in 1..ncells, j in 1..npcells}:
        x_ls[i,j] >= lb[i];
subj to QR2{i in 1..ncells, j in 1..npcells}:
        x_ls[i,j] <= ub[i];
subj to QR5{j in 1..npcells}:
        x_ls[p[j],j] <= a[p[j]] - plpl[j];
subj to QR6 {i in 1..nconstraints, j in 1..npcells}:
        sum {t in begconst[i]..begconst[i+1]-1} coef[t]*x_ls[xcoef[t],j] = b[i];

subj to QRu1{i in 1..ncells, j in 1..npcells}:
        x_us[i,j] >= lb[i];
subj to QRu2{i in 1..ncells, j in 1..npcells}:
        x_us[i,j] <= ub[i];
subj to QRu5{j in 1..npcells}:
        x_us[p[j],j] >= a[p[j]] + pupl[j];
subj to QRu6 {i in 1..nconstraints, j in 1..npcells}:
        sum {t in begconst[i]..begconst[i+1]-1} coef[t]*x_us[xcoef[t],j] = b[i];
```

- IP.run

```
reset;

model directe.mod;
data simple.dat;

option solver cplexamp;
option omit_zero_rows 1;
option display_eps .000001;
option cplex_options $cplex_options 'presolve= 0';

problem Master: lb,ub,x_ls, x_us, R1, R2, R3, R4, QR1, QR2, QR5, QR6, QRu1, QRu2, QRu5,
QRu6, BPr;

#########################################################
#Reading of data and initializations #
#########################################################

printf"\n Size of the problem \n\n" >output.txt;
printf"\t Number of cells = %d\n",ncells >output.txt;
printf"\t Number of sensible cells = %d\n",npcells >output.txt;
printf"\t Number of constraints = %d\n",nconstraints >output.txt;
printf"\t Numero of non-zeros in the constraints = %d\n",nnz >output.txt;

#####################################################
# Solving problem #
#####################################################

problem Master;
option cplex_options 'presolve=0 mipdisplay=1 timing=1 feasibility=1.0e-9 integrality=0
primalopt';
solve Master;
printf "\n FINAL SOLUTION\n\n";
for{i in 1..ncells}
{
display lb[i];
display ub[i];
}
```

Here follows the AMPL code for the auditing of the solutions.

- auditing.mod

```
################################################
# PARAMETERS
################################################

param ncells integer >0; # number of cells

param npcells integer>0; # number of sensible cells

param nnz integer >0;

param nconstraints integer >0; # number of restrictions (m)

param c {1..ncells} ; # weights of the cells (w_i)

param is_p {1..ncells}; # 1 if the cell is sensible, 0 otherwise

param a {1..ncells}; # original values of the cells
param lb {1..ncells}; # lower bounds for increments/decrements
param ub {1..ncells}; # upper bounds for increments/decrements
param b {1..nconstraints}; # right hand side for constraints

param begconst{1..nconstraints+1}; #pointer to begin info. in coef and xcoef
param coef{1..nnz}; #constraints coefficients
param xcoef{1..nnz}; #index of variable affected for each coefficient

param p {1..npcells}; # position

param s;

param Ep default .000001; #maximum relative deviation allowed



################################################
# Definition of Q_l,s #
################################################

var x_ls{1..ncells};

minimize BQ_ls:
        x_ls[p[s]];
subj to QR1{i in 1..ncells}:
        lb[i] <= x_ls[i];
subj to QR2{i in 1..ncells}:
        x_ls[i] <= ub[i];
subj to QR3 {i in 1..nconstraints}:
        sum {t in begconst[i]..begconst[i+1]-1} coef[t]*x_ls[xcoef[t]] = b[i];
```

```
#########################################################
# Definition of Q_u,s #
#########################################################

var x_us{1..ncells};

maximize BQ_us:
        x_us[p[s]];
subj to QRu1{i in 1..ncells}:
        lb[i] <= x_us[i];
subj to QRu2{i in 1..ncells}:
        x_us[i] <= ub[i];
subj to QRu3 {i in 1..nconstraints}:
        sum {t in begconst[i]..begconst[i+1]-1} coef[t]*x_us[xcoef[t]] = b[i];
```

- auditing.run

```
reset;

model auditing.mod;
data auditing2.dat;

option solver cplexamp;
option omit_zero_rows 1;
option display_eps .000001;


problem Superior: x_ls, QR1, QR2, QR3, BQ_ls;
problem Inferior: x_us, QRu1, QRu2, QRu3, BQ_us;

#####################################################
#Reading of data and initializations # #
#####################################################

printf"\n Size of the problem \n\n" >output.txt;
printf"\t Number of cells = %d\n",ncells >output.txt;
printf"\t Number of sensible cells = %d\n",npcells >output.txt;
printf"\t Number of constraints = %d\n",nconstraints >output.txt;
printf"\t Numero of non-zeros in the constraints = %d\n",nnz >output.txt;
#####################################################
#Solving #
#####################################################

for {s2 in 1..npcells}
{
	printf "\nITERATION %d\n\n", s2;
	#Recalculate parameters
	let s := s2;
	problem Superior;
	option cplex_options 'presolve=0 mipdisplay=1 timing=1 feasibility=1.0e-9 integrality=0
primalopt';
	solve Superior;
	display x_ls[p[s]];
	problem Inferior;
	solve Inferior;
	display x_us[p[s]];

};
```

References

[1] Robert Fourer, David M. Gay and Brian W. Kernighan, AMPL: A Modeling Language for Mathematical Programming, Second edition, (2003).

[2] Fred Glover, Lawrence H. Cox and Rahul Patil, Integrated exact, hybrid and metaheuristic learning methods for confidentiality protection. Springer (2011).

[3] J.F. Benders, Partitioning procedures for solving mixed-variables programming problems, Computational Management Science, 2005, vol 2, 3-19. English translation of the original paper appeared in Numerische Mathematik, 4:238–252 (1962).

[4] Dimitris Bertsimas and John Tsitsiklis, Introduction to linear optimization. Belmont: Athena Scientifc (1997).

[5] J. Castro, A. Frangioni and C. Gentile, Perspective reformulations of the CTA problem with L2 distances, Operations Research, 62(4) (2014) 891-909.

[6] J. Castro, On assessing the disclosure risk of controlled adjustment methods for statistical tabular data, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 20 (2012) 921-941.

[7] J. Castro, Recent advances in optimization techniques for statistical tabular data protection, European Journal of Operational Research, 216 (2012) 257-269.

[8] J. Castro, Minimum-distance controlled perturbation methods for large-scale tabular data protection, European Journal of Operational Research, 171 (2006) 39-52.

[9] J. Castro and D. Baena, Using a Mathematical Programming Modeling Language for Optimal CTA, Lecture Notes in Computer Science, 5262 (2008) 1-12, eds. J. Domingo-Ferrer and Y. Saigin, Springer.